



# Optimization of the structural complexity of artificial neural network for hardware-driven neuromorphic computing application

Kannan Udaya Mohanan<sup>1</sup> · Seongjae Cho<sup>1</sup> · Byung-Gook Park<sup>2</sup>

Accepted: 17 May 2022 / Published online: 8 July 2022  
© The Author(s) 2022

## Abstract

This work focuses on the optimization of the structural complexity of a single-layer feedforward neural network (SLFN) for neuromorphic hardware implementation. The singular value decomposition (SVD) method is used for the determination of the effective number of neurons in the hidden layer for Modified National Institute of Standards and Technology (MNIST) dataset classification. The proposed method is also verified on a SLFN using weights derived from a synaptic transistor device. The effectiveness of this methodology in estimating the reduced number of neurons in the hidden layer makes this method highly useful in optimizing complex neural network architectures for their hardware realization.

**Keywords** Neural networks · Hardware neuromorphic systems · Synaptic device · Neuron circuits · Hidden layer · Pattern recognition · Singular value decomposition (SVD)

## 1 Introduction

Capability of the human brain to process data in a parallel and energy-efficient architecture is driven by its huge network of  $\sim 10^{11}$  neurons and  $\sim 10^{15}$  synapses. The biological neuron is the fundamental processing unit inside the brain. A neuron consists of a cell body which is called as the soma to which the axons and dendrites are attached (Fig. 1a). The neuron receives information through the dendrites from the pre-synaptic neuron and the information is passed on to the post-synaptic neuron through the axon using action potentials. These signals arrive at the junction between two adjacent neurons which are referred to as the synaptic cleft. These synapses are capable of processing and memorizing the information signal. The connection strength between the two adjacent neurons is determined by the synaptic weight. Analogous to the biological neural

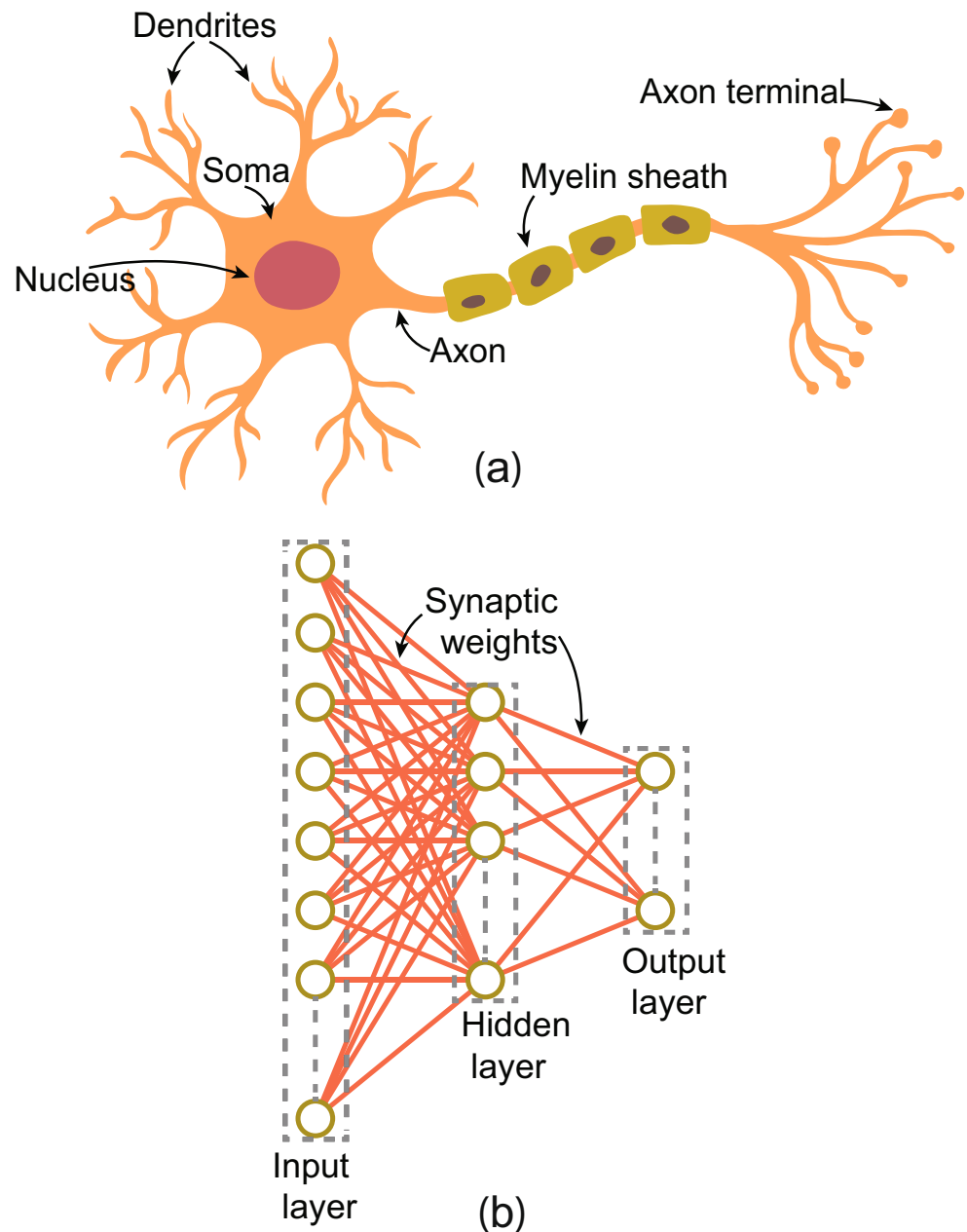
network, researchers have tried to develop hardware neuromorphic systems [1, 2] which emulate the working of the human brain. These are purely electrical circuits composed of neuron circuitry [3–5] and the associated synaptic array elements. Such a bio-inspired approach can overcome several of the shortcomings of current computer architectures which require large amount of data transfer between the processing and memory units leading to an increased energy expenditure and computational times. Currently, software-based neural network designs are also in much demand due to their capability to solve data-intensive computational tasks like face recognition, self-driving cars, and big data analytics. A feedforward neural network (Fig. 1b) is an artificial neural network (ANN) which is capable of solving several tasks like pattern recognition, prediction, and function approximation. These artificial neural networks are modelled on the biological neural circuitry and are primarily composed of neurons and synapses. While the synapses are responsible for the storage of information called as the ‘weights’, the neurons implement the processing capabilities of the network using vector matrix multiplication (VMM) [6] followed by input summation and threshold firing. Non-linear mathematical functions called ‘activation functions’ are used for determining the output of the neurons from the integrated input signals. The rectified linear unit (ReLU) [7] function is a popular choice for the neuron activation function because of its similarity to biological neurons in zeroing the negative weight values. In an ANN,

✉ Seongjae Cho  
felixcho@gachon.ac.kr

<sup>1</sup> Graduate School of IT Convergence Engineering, Gachon University 1342 Seongnamdaero, Sujeong-gu, Seongnam-si, Gyeonggi-do, 13120 Republic of Korea

<sup>2</sup> Department of Electrical and Computer Engineering, Seoul National University Gwanak-ro 1, Gwanak-gu, Seoul 08826, Republic of Korea

**Fig. 1** Schematics of (a) biological neuron and (b) single-layer feed forward neural network (SLFN)



these series processes of summing the inputs and computing the VMM followed by the non-linear activation of the neurons are carried forward from the input to the next hidden layer and finally to the output neurons. This process is referred to as the “feedforward” process, and hence, the name of the network is usually called as “feed-forward neural network”. Once the output values are obtained from the final layer neurons, they are compared with the actual values and the difference between the two values is computed using a loss function. The mean cross-entropy loss (MCEL) function is a commonly-used loss function in classification tasks which compare the predicted and actual probabilities of the output. The gradient of the loss function is then computed

(referred to as “stochastic gradient descent”) with respect to the weights of each layer and the weights are updated based on the direction of steepest decrease of the gradients. This backward propagation of the errors from the final output layer to the initial layers is called the “back-propagation” method. The “learning rate” determines the magnitude by which the weights are updated at every back-propagation step. A complete cycle of feedforward and backpropagation is computed over a batch of input images called the batch size. For a typical pattern recognition task, the whole dataset is divided into several smaller batches of images with equal batch sizes for the ANN training. Once the training process is carried out over an entire dataset, it is referred to as one “epoch”

of training. The same procedure is followed several times through multiple epochs until the mean accuracy and MCEL function converges to a satisfactory value. An ANN network can have multiple layers of neurons between its input and output neurons, depending on the complexity of the problem at hand. A single-layer feedforward neural network (SLFN) is an ANN with a single hidden layer of neurons and is the most commonly used ANN for pattern recognition tasks. Such a neural network can easily be adopted into a hardware synaptic array along with neuron circuits for the realization of hardware neuromorphic system. Hence, optimizing the structural complexity of feedforward neural networks is highly important from a hardware perspective.

### 1.1 Motivation

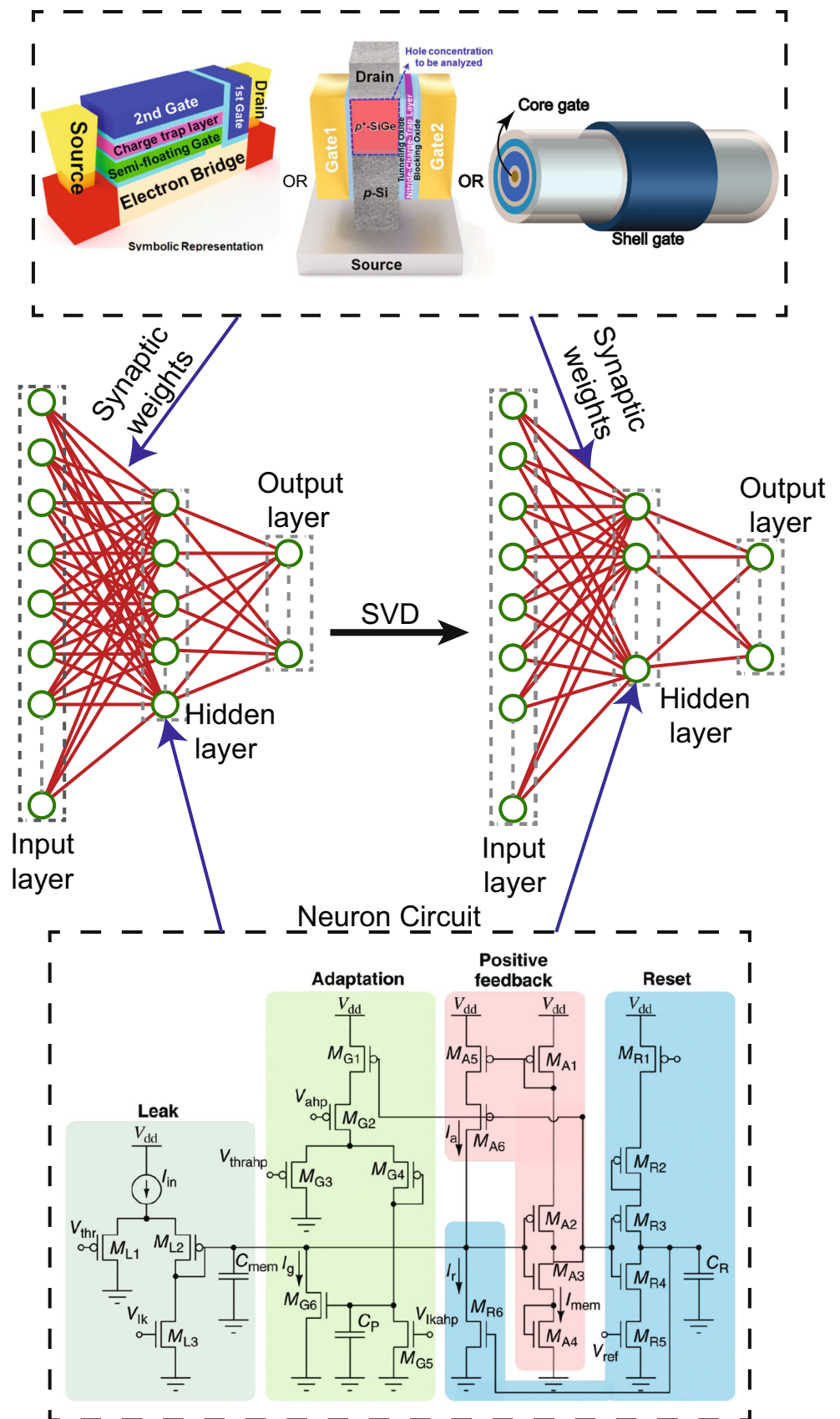
There has been a consistent effort from engineers to realise neuromorphic architecture in hardware sense. Neuromorphic hardware design pursues an in-memory computing with highly energy-efficient parallel processing capability in the end. Realizing the artificial neural network through complementary metal-oxide-semiconductor (CMOS) integrated circuits is the first step in this direction. IBM's TrueNorth chip has been one of the first major breakthroughs in this respect [8]. This chip encompasses 256 million synapses and 1 million integrate-and-fire (IF) neurons fabricated using the 40-nm-node CMOS technology. Similarly, Intel's Loihi chip was fabricated with 130 million synapses and 131,072 neurons utilizing 14-nm-node CMOS technology. Both chips support the high capacity and accelerated learning in the specifically designed neuromorphic semiconductor chip. Several other neuromorphic chips have been reported: Neurogrid [9], Braindrop [10], BrainScales [11], SpiNNaker [12], ROLLS [13], Darwin [14], DynapSEL [15] etc. Synaptic transistors construct the core of these neuromorphic chips wherein the weight information is stored. Unlike software-based weight initialization, the synaptic devices can only have limited number of conductance values (weights). More recently, the synaptic devices are realized by a single memory device including charge-trap flash (CTF) memory [16], resistive-switching random-access memory (RRAM) [6, 17–20], phase-change random-access memory (PRAM) [21], ferroelectric random-access memory (FRAM) [22], and magnetic random-access memory (MRAM) [23] for increasing the synapse array density and minimizing the power consumption, departing from the conventional synapses made up of circuits or several electron devices. Also, many of today's state-of-the-art deep learning models such as Inception v1, VGG-19, ResNet, and others contain huge number of neurons with millions of parameters [24–27]. These are extremely hard to implement in the hardware due to the limitations with array size

constraints (chip compactness) and manufacturing cost. For the widespread adoption of these models in the neuromorphic hardware, their model size needs to be reduced and optimized with the hardware constraints (see Fig. 2). It is in this context that we have verified our current methodology for the optimized SLFN design using the weights extracted from a synaptic transistor device [28]. To the best of our knowledge, this is the first work focusing on the optimization of SLFN for implementation of the specifically designed artificial intelligence chip.

### 1.2 Literature review

A major reason for the structural complexity of a neural network is the number of neurons in the hidden layer used for training the data. Several research groups have attempted to reduce the structural complexity of neural networks. The seminal work by Arai resulted in a hyperplane-based algorithm for estimating the hidden neuron number and found that  $I - 1$  hidden units are sufficient for  $I$  learning patterns [29]. Later, Tamura and Tateishi proposed a method to determine the number of hidden neurons using information criteria [30]. Similarly, Fujita et al., statistically estimated the hidden neuron number with the advantage of fast learning [31]. The required neuron number is given by  $N_h = K \log ||P_c Z|| / \log S$ , where  $S$  is defined as the number of candidates which are arbitrarily searched for optimised hidden neurons and  $c$  is the acceptable error limit. E. J. Teoh et al. used an SVD-based approach to estimate the number of hidden neurons in a feed-forward network [32]. But the method was empirical in nature with results shown on small datasets only. Pruning is another well-known method for reducing neural network size by removing structural parameters based on the relevance scores. Recently, F.E Fernandes et al. used a filter-pruning strategy followed by fine tuning to reduce the size of some popular models like ResNets, DenseNets etc [33]. There have also been several other strategies adapted for pruning and is the most widely used method for model compression [34, 35]. However, the method often requires several iterations and fine-tuning making it hard to adapt to hardware implementations. There have also been several advanced techniques like quantization [36], Huffman encoding [37], and neural architecture search (NAS) for determining the optimal neural network size [38]. However, most of these algorithms are developed for efficient implementation on software using the conventional von Neumann hardware architecture. They require high computing power using advanced hardware such as GPU or cloud-based computational units. A major limitation of such methods is encountered in their implementation for neuromorphic hardware where these algorithms have never been tested with realistic physical constraints.

**Fig. 2** Schematic showing the optimization of an SLFN using SVD method. In the hardware sense, the synaptic weights can be obtained from a single memory component, for example, a 4-terminal synaptic transistor [28, 61, 62] whereas the neurons in the hidden layer are to be realized by relatively bulky CMOS circuits which consume large area and energy [40]. The inset shows a neuron circuit consisting of two n-type metal-oxide-semiconductor field-effect transistors (NMOSFETs) and passive devices ( $C_{mem}$ : membrane capacitor,  $V_{mem}$ : membrane potential). Adapted from [41] with permission from Wiley Books. The SVD approach is highly practical in optimizing the hardware neural network from the design level determining the number of neuron circuits and the size of synapse array



### 1.3 Contribution

The main goal of this work is to optimise the structural complexity of a SLFN for hardware realisation. We propose the use of SVD method for determining the optimized number of neurons in the hidden layer for a SLFN. SVD is a comparably simpler method and require much less iterations to identify the lowest possible model size within an acceptable error limit. Unlike other reported algorithms, we have verified our methodology by incorporating physically reliable weight values from a core-shell dual-gate (CSDG) nanowire transistor [28] device for identifying the convergence criterion. Such a device-based verification is important from a hardware perspective. Through the device-based weights and weight level variations, we have taken into account the hardware limitations like limited range of device conductance values and device non-linearity considerations while defining the convergence criterion for determining the reduced number of neurons. To the best of our knowledge, this is the first work focusing on the optimization of SLFN for hardware implementation. We believe that this is a very important step in realizing compact neural networks for hardware realization.

## 2 Mathematical preliminaries

We consider a general SLFN consisting of  $J$  input neurons,  $n$  neurons in the hidden layer, and  $O$  output neurons. The SLFN is trained using a set of  $T$  training images  $X \in \{X_1, X_2, \dots, X_T\}$ . The activation function used is the rectified linear unit (ReLU) function. The trained model is then used to recognize a set of  $M$  test images. The activation matrix is given by the matrix  $K$  such that  $K \in \mathbb{R}^{M \times n}$ . The rows of  $K$  can be considered as  $M$  data points in an  $n$  dimensional space since  $K$  is an  $M \times n$  matrix. From a hardware perspective, realization of such an SLFN would require synaptic devices like resistive-switching random-access memory [17], charge-trap flash memory [16], field-effect transistor [39], phase-change memory [21] for the storage of weight values cooperating with CMOS neuron circuits [40, 41]. Essentially, the synaptic devices utilize their electrical conductance states as the equivalent weight values and the neuron circuits are operated based on the behavior of the activation function. Although the majority of area and power efficiencies are determined by the high-density synaptic device array in the hardware neural network, while a synaptic device can be realized by a single electron device with very high scalability reaching down to nanometer dimension, a neuron is implemented in the form of complementary metal-oxide-semiconductor (CMOS) integrated circuit based on plural passive and

active components, and thus, requires relatively larger area and power consumption compared with a single synapse. For this reason, the mathematical approach is focused on reducing the number of neurons, which has not been seriously considered in the conventional purely software-oriented artificial intelligence approach. Hence, optimally minimizing the number of neurons in the hidden layer should be one of the major concerns for the neuromorphic hardware implementation. In the current work, we use the SVD method to reduce the size of the activation matrix  $K$  without compromising on the prediction capabilities of the network. This is done by determining the best-fitting lower dimensional subspace for the set of  $M$  data points. For the activation matrix  $K$ ;  $\mathbf{s}_{L_1}, \mathbf{s}_{L_2}, \mathbf{s}_{L_3}, \dots, \mathbf{s}_{L_n}$  are the left singular vectors;  $\mathbf{s}_{R_1}, \mathbf{s}_{R_2}, \mathbf{s}_{R_3}, \dots, \mathbf{s}_{R_n}$  are the right singular vectors and  $\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n$  are the singular values. The left singular vectors:

$$\mathbf{s}_{L_i} = \frac{1}{\sigma_i} \mathbf{K} \mathbf{s}_{R_i} \quad (1)$$

where the coordinates of  $\sigma_i \mathbf{s}_{L_i}$  are the projections of rows of  $K$  onto  $\mathbf{s}_{R_i}$ . Essentially, the singular vectors are the orthonormal basis for the lower-dimensional subspaces and the singular values are the projections of the  $M$  data points onto these subspaces.

The SVD of the matrix  $K$  is the factorization of the matrix into the product of three separate matrices  $S_L \in \mathbb{R}^{M \times M}$ ,  $\beta \in \mathbb{R}^{M \times n}$ , and  $S_R \in \mathbb{R}^{n \times n}$  such that:

$$K = S_L \beta S_R^T \quad (2)$$

$$\text{where } S_L = \begin{bmatrix} S_{L11} & \dots & S_{L1M} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ S_{LM1} & \dots & S_{LMM} \end{bmatrix}_{M \times M},$$

$$S_R = \begin{bmatrix} S_{R11} & \dots & S_{R1n} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ S_{Rn1} & \dots & S_{Rnn} \end{bmatrix}_{n \times n} \quad \text{and } \beta = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots \\ \vdots & \sigma_n & \dots & \dots \\ \vdots & \dots & \dots & \dots \\ 0 & \dots & \dots & \sigma_M \end{bmatrix}_{M \times n}$$

The columns of  $S_L$  and  $S_R$  are the left and right singular vectors of the  $K$  matrix, respectively, and are orthonormal to each other.  $\beta$  is a diagonal matrix with  $n$  non-zero diagonal elements arranged such that  $\sigma_1 \geq \sigma_2 \geq \sigma_3, \dots \geq \sigma_n \geq 0$ . Further,  $\sigma_{n+1} = \sigma_{n+2} = \dots = \sigma_M = 0$ . Here, the matrix  $K$  can be expressed as  $K = \sum_{i=1}^n \sigma_i \mathbf{s}_{L_i} \mathbf{s}_{R_i}^T$  which can be proved by the following theorem [42]:

**Theorem 1** For an  $M \times n$  matrix  $K$ , with right singular vectors  $\mathbf{s}_{R_1}, \mathbf{s}_{R_2}, \mathbf{s}_{R_3}, \dots, \mathbf{s}_{R_n}$ ; left singular vectors

$\mathbf{S}_{L_1}, \mathbf{S}_{L_2}, \mathbf{S}_{L_3}, \dots, \mathbf{S}_{L_n}$  and corresponding singular values  $\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n$ .

$$K = \sum_{i=1}^n \sigma_i \mathbf{S}_{L_i} \mathbf{S}_{R_i}^T \tag{3}$$

*Proof* We see that multiplying the left-hand side of (3), i.e. matrix  $K$  with  $\mathbf{S}_{R_j}$  yields  $\sigma_j \mathbf{S}_{L_j}$  (from (1)). Similarly, multiplying the right-hand side of (3), i.e.  $\sum_{i=1}^n \sigma_i \mathbf{S}_{L_i} \mathbf{S}_{R_i}^T$  with  $\mathbf{S}_{R_j}$  yields  $\sigma_j \mathbf{S}_{L_j}$  since  $\mathbf{S}_{R_i}$  are orthogonal to each other and  $\mathbf{S}_{R_i}^T \times \mathbf{S}_{R_i} = I$  where  $I$  is the identity matrix.

$$K \mathbf{S}_{R_j} = \sigma_j \mathbf{S}_{L_j} = \sum_{i=1}^n \sigma_i \mathbf{S}_{L_i} \mathbf{S}_{R_i}^T \mathbf{S}_{R_j} \tag{4}$$

Any two matrices  $A$  and  $B$  are equal if  $Ax = Bx$  for all vectors  $x$ . Hence, from (4),

$$K = \sum_{i=1}^n \sigma_i \mathbf{S}_{L_i} \mathbf{S}_{R_i}^T$$

□

### 3 Reducing the over-estimated rank $n$

Most often, the number of neurons in the hidden layer taken into account for the SLFN training is highly over-estimated. Such an over-estimation leads to additional neuronal circuitry in physical cross-bar array implementations, which can be avoided by determining the reduced rank. Hence, the reduction of the over-estimated rank  $n$  is highly significant from a CMOS manufacturing perspective. In the mathematical sense, the over-estimation of neurons in the hidden layer of SLFN leads to the creation of additional hyperplanes in the hidden layer activation space. Such hyperplanes are essentially parallel planes to the linearly independent basis vectors of the space. An intuitive way of dealing with this problem was put forward by G.W. Stewart, where the excess neurons are considered as “contaminations” to the original matrix [43]. Now, the problem reduces to the determination of the reduced rank  $g$  of the hidden-layer activation matrix from the over-estimated matrix with  $n$  neurons. The sum of squares of all the entries in a matrix is given by the Frobenius norm of the matrix.

$$\|K\|_F = \sqrt{\sum_{j,i} k_{j,i}^2} \tag{5}$$

For any row,  $k_j$  of matrix  $K$ , we can show that  $\sum_{j=1}^M |k_j|^2 = \sum_{i=1}^n \sigma_i^2$  [42]. Hence from (5), we have:

$$\|K\|_F^2 = \sum_{i=1}^n \sigma_i^2 \tag{6}$$

Thus, the square of the norm of the activation matrix is equal to the sum of the squares of the singular values of  $K$ . If we assume  $E$  as the “contamination matrix” which accounts for the excess neurons in the  $K$  matrix, then from (6),

$$\sigma_{g+1}^2 + \sigma_{g+2}^2 + \dots + \sigma_n^2 \leq \|E\|_F^2 \tag{7}$$

Therefore, if  $\sigma_g$  is sufficiently large in comparison to  $E$ , then SVD can provide a reasonable approximation of the subspace spanned by the hidden-layer activation matrix.

### 3.1 Determination of $g$

Several different criteria have been previously suggested for the determination of  $g$  [44], within a threshold value  $\gamma$  such that the “spectral energy” of  $K$  matrix is retained. We use the convergence criterion for  $g$  based on the spectral energy approximation as shown below:

$$\sum_{i=1}^g \sigma_i^2 / \sum_{i=1}^n \sigma_i^2 \geq \gamma \tag{8}$$

$\gamma$  has been mostly determined by visual inspection based on the problem in previous works. Although  $\gamma$  originates from the spectral energy equation in (8), it has actual physical significance when dealing with real hardware-based systems.  $\gamma$  has a direct dependence on the final accuracy of the trained model and hence the determination of  $\gamma$  should take into account the device considerations like the number of stable weights attainable by the synaptic device. Unlike the near-continuous weights available for a software based artificial neural network, realistic hardware implementations can only have a finite number of stable conductance levels [45] which can be utilized for the synaptic array fabrication. Such practical considerations are extremely important in the determination of  $\gamma$  since the activation matrix output might alter significantly with a finite-weight network and so does the convergence criterion for the SVD method. Hence, we have taken into consideration the various practical weight levels ranging from 2, 4, 8, 16, and 32 levels for the application of the SVD method in optimizing the number of neurons in the hidden layer.

## 4 Results & discussion

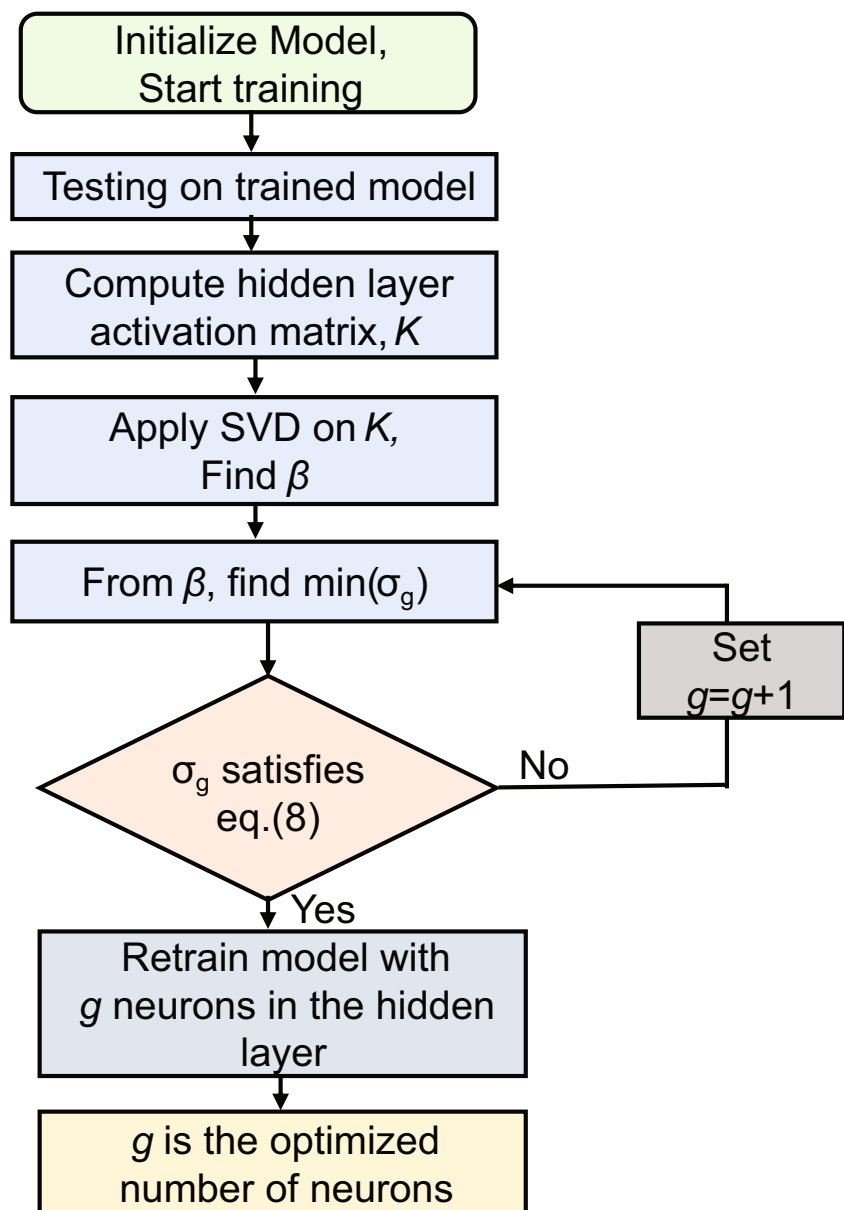
The proposed method was applied to the Modified National Institute of Standards and Technology database (MNIST) dataset using two types of weight initializations. In the first case, weights were randomly initialised using a purely software-based randomization technique. In the second case, we have utilised discrete synaptic weights extracted from a core-shell dual-gate nanowire synaptic transistor [28].

#### 4.1 Software-based weight initialization

The MNIST dataset consists of hand-written digits of 60,000 training images and 10,000 test images [46]. The images correspond to digits from “0” to “9” with nearly uniform distribution among all the 10-digit classes. Every image is in grayscale format with a resolution of  $28 \times 28$  pixels. Before training, the input data was normalized to a range of  $[0, 1]$ , and then, reshaped to form a column matrix of 784 rows. The normalized data is then fed to a SLFN with 784 input neurons, 100 neurons in the hidden layer, and 10 output neurons. All the simulations were performed using PyTorch open source package [47]. The input data are trained for 100 epochs within which the mean classification error was reduced to a stable value. The

detailed flowchart and algorithm used for optimization of the SLFN are shown in Fig. 3 and Algorithm 1, respectively. Briefly, the training of the SLFN for the input data is initialized using a large estimate for the neurons in the hidden layer, say 100. The model is then trained for all the images in the training dataset using the rectified linear unit (ReLU) as the non-linear activation function for the SLFN. The back-propagation algorithm is implemented using the stochastic gradient descent algorithm. From the trained model, the accuracy is determined using the test dataset. The SVD of the hidden layer activation output is then computed and the  $\beta$  matrix is found. Using a  $\gamma$  value of 0.97, a reduced rank value of 42 is determined for  $g$  such that the singular value  $\sigma_g$  satisfies the threshold criterion in (8). The SLFN training procedure is then repeated for 42

**Fig. 3** Flowchart describing the execution of the proposed SVD algorithm

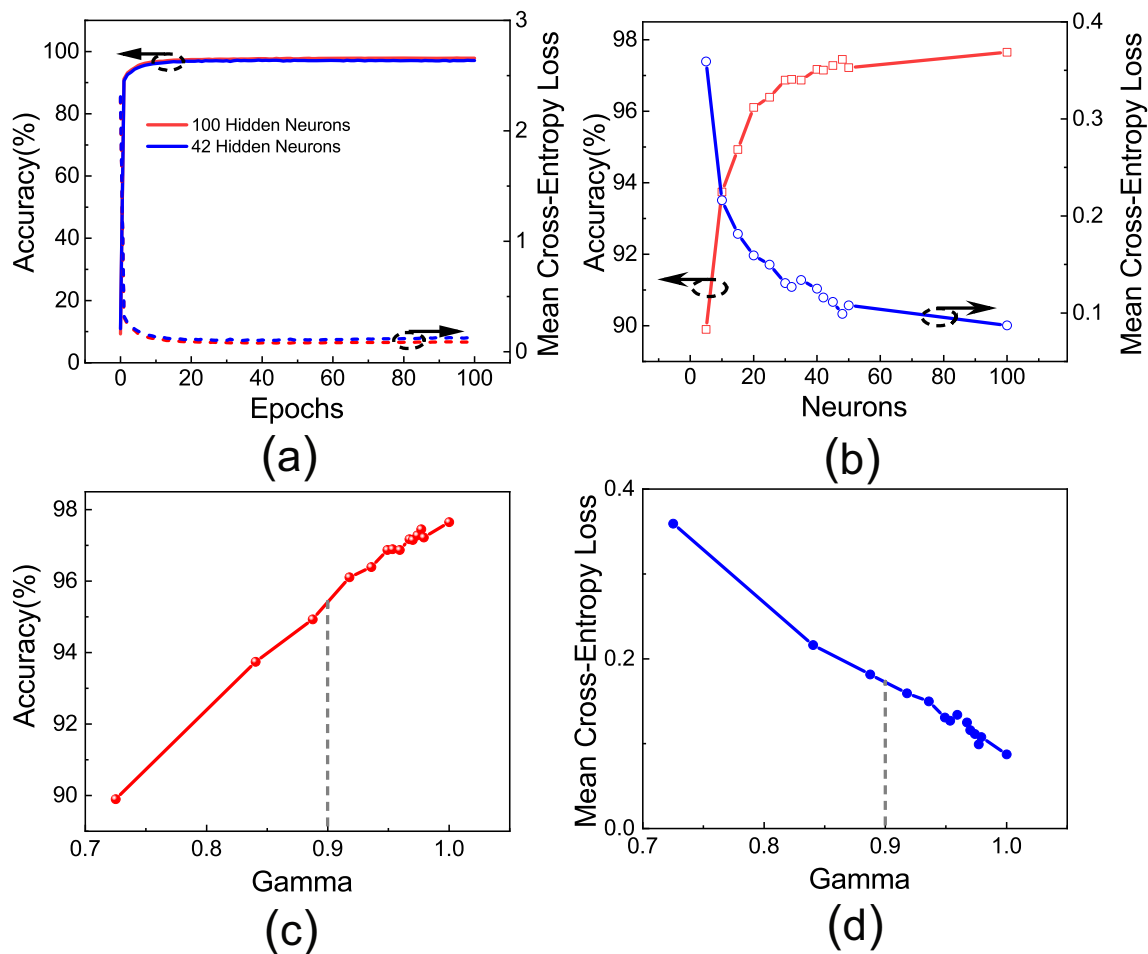


neurons in the hidden layer. Figure 4a shows the variation of digit recognition accuracy with the number of epochs for the network with both 100 and 42 neurons in the hidden layer. We obtained a maximum test accuracy of 97.65% for 100 neurons in the hidden layer and 97.15% for 42 neurons in the hidden layer. From this, it is evident that the test accuracy does not increase appreciably with the addition of neurons beyond 42 in the hidden layer. Similarly, the decrease of MCEL function also follows a similar trend for both 100 and 42 neurons in the hidden layer. The accuracy and the loss function do not vary significantly beyond 42 neurons in the hidden layer since this is the reduced rank representation of the activation matrix space. Further addition of neurons results in the creation of redundant neurons which adversely affects the network since it adds only to the structural complexity of the network leading to increased computational time.

The variation of test accuracy and MCEL function as a function of the number of neurons in the hidden layer

is shown in Fig. 4b. It is observed from the figure that the accuracy does not increase considerably beyond 42 neurons in the hidden layer. As discussed before, the determination of a lower bound for  $\gamma$  is crucial for practical implementation of the SVD method for rank reduction. Figure 4c and d show the dependences of accuracy and MCEL function on the value of  $\gamma$ . We can infer that a value of 0.9 is a safe choice for the lower bound on  $\gamma$  since the percentage loss in accuracy between  $\gamma = 1$  and  $\gamma = 0.9$  is below 2.3%. Similarly, the MCEL function also does not increase significantly until about  $\gamma = 0.9$ .

Apart from the recognition accuracy, an important performance metric of a neural network model is its capability to predict all the output classes with a similar level of confidence. The confusion matrix (see Fig. 5) can be used to measure the overall prediction capabilities of a model. It is a square matrix which compares the true output values with that predicted by the model. The diagonal elements represent the cases where the predicted output



**Fig. 4** MNIST digit recognition accuracy (%) and mean cross-entropy loss as a function of (a) number of training epochs for 100 and 42 neurons in the hidden layer. (b) Test accuracy (%) and MCEL variation with neurons. (c) Accuracy and (d) mean cross-entropy loss variation with gamma



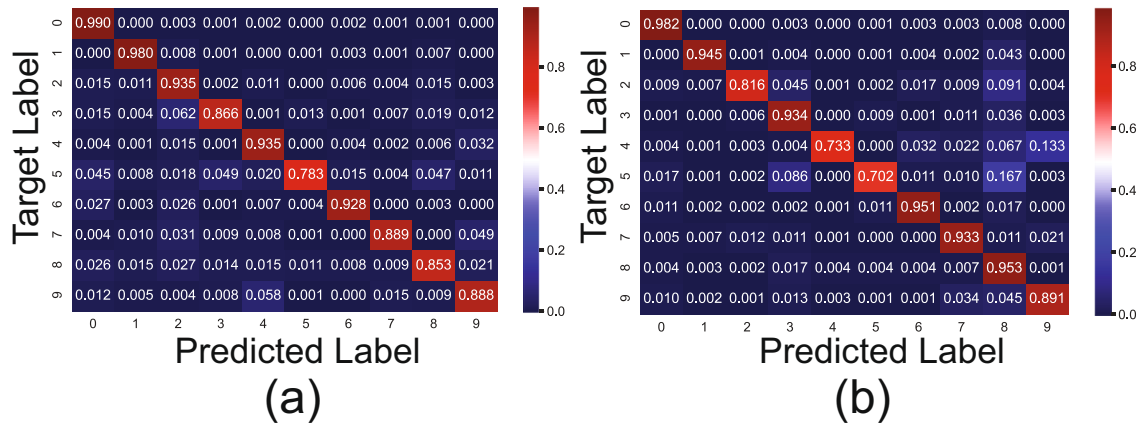


Fig. 5 Confusion matrix for the MNIST test data for (a) 100 and (b) 42 neurons in the hidden layer

is same as the true output. All the off-diagonal elements represent the faulty predictions made by the model. From Fig. 5a and b we see that our trained model is showing a high degree of predictability for all the classes even with

a reduced number of neurons. This indicates that the SVD method has reduced the effective ranks of the activation matrix without compromising the class separability in model prediction. In order to evaluate the performance of

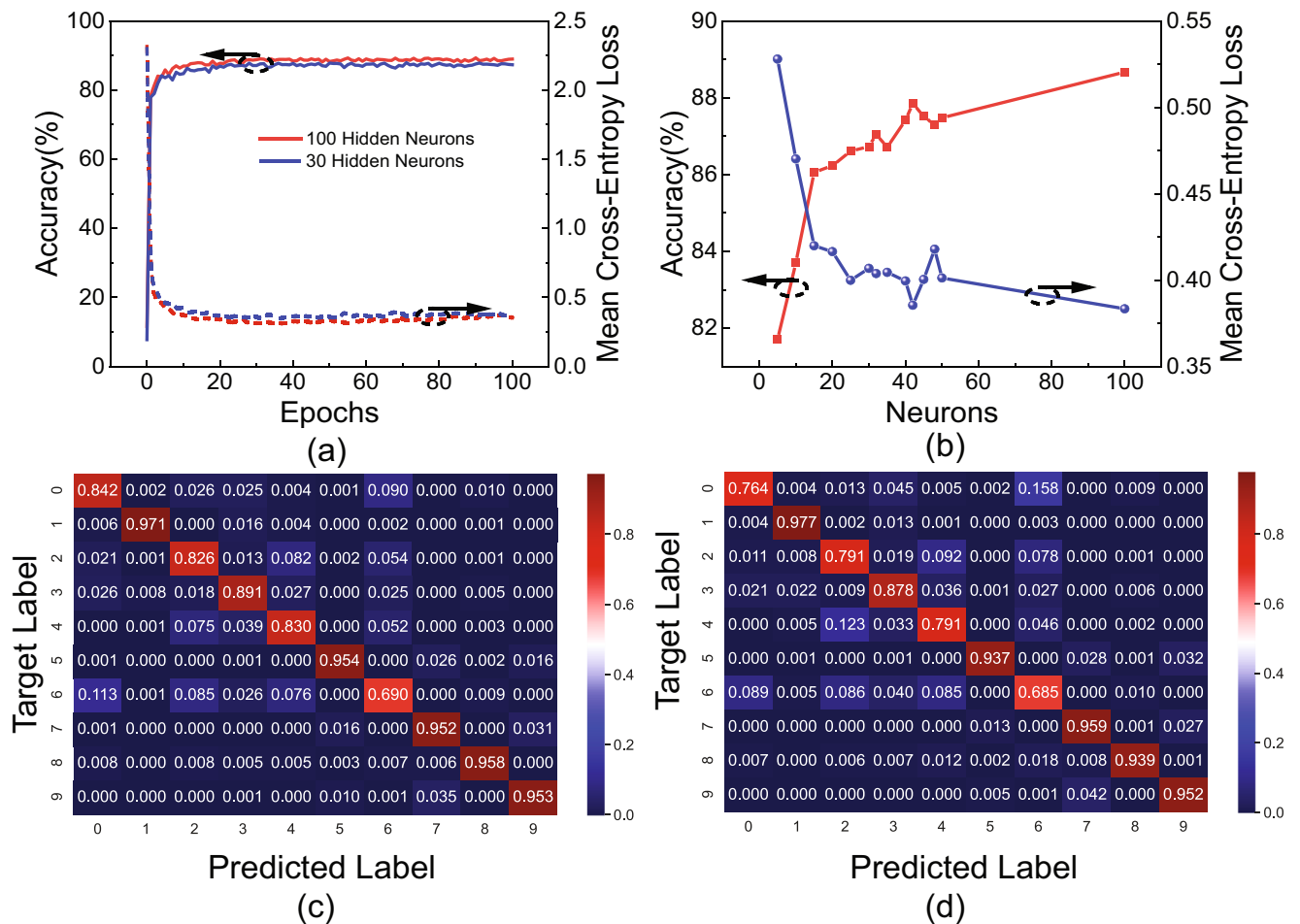


Fig. 6 Application of the proposed method to fashion MNIST pattern recognition. (a) Recognition accuracy (solid) and mean cross-entropy loss (MCEL) (dotted) as a function of number of training epochs with

100 and 30 neurons in the hidden layers. (b) Recognition accuracy (red) and MCEL (blue) as a function of neurons in the hidden layer. Confusion matrices for (c) 100 and (d) 30 neurones in the hidden layer

our SVD method on other datasets, we have applied our proposed method on two more datasets – fashion MNIST and Street View House Numbers (SVHN) Datasets [48, 49]. fashion MNIST dataset is similar to the MNIST dataset except for the type of images being classified. The dataset contains greyscale images ( $28 \times 28$ ) of fashion articles which are to be classified into 10 output classes. However, the SVHN dataset consists of RGB images of  $32 \times 32$  resolution consisting of real word images of house numbers from Google Street View images. The dataset is to be classified into 10 output classes corresponding to the images of the digits at the center of the house number. Figure 6a shows the application of the proposed method in reducing the number of neurons in the hidden layer from 100 down to 30 neurons for fashion MNIST pattern recognition tasks. It is observed that, as the number of neurons decreases from 100 down to 30, there is only a marginal drop in test accuracy from 88.67% to 86.73%. Similarly, the MCEL shows only a small drop by the scheme with reduced models. Here, it is evident

that the SVD method succeeds in removing redundant neurons from the hidden layer without an appreciable drop in network performances. Also, the variation in test accuracy (%) with the number of neurons in the hidden layer depicted in Fig. 6b reveals that the model performance is not improved with further decrease in neuron number below a cut-off value of 30. The confusion matrices shown in Fig. 6c and d correspond to the SLFNs with 100 and 30 neurons in the hidden layer, respectively. It is shown that the lowering of model complexity depending on application of the SVD method has not affected the capability of the model to make distinctions among the various output classes. Figure 7a shows the SVHN digit recognition accuracy and MCEL as a function of number of training epochs with 100 and 48 neurons in the hidden layer. As compared with the test results from MNIST and fashion MNIST datasets, one from the SVHN dataset show large fluctuations in both accuracy and MCEL over the training process. The saturated accuracy value is lower than those obtained from the previous MNIST

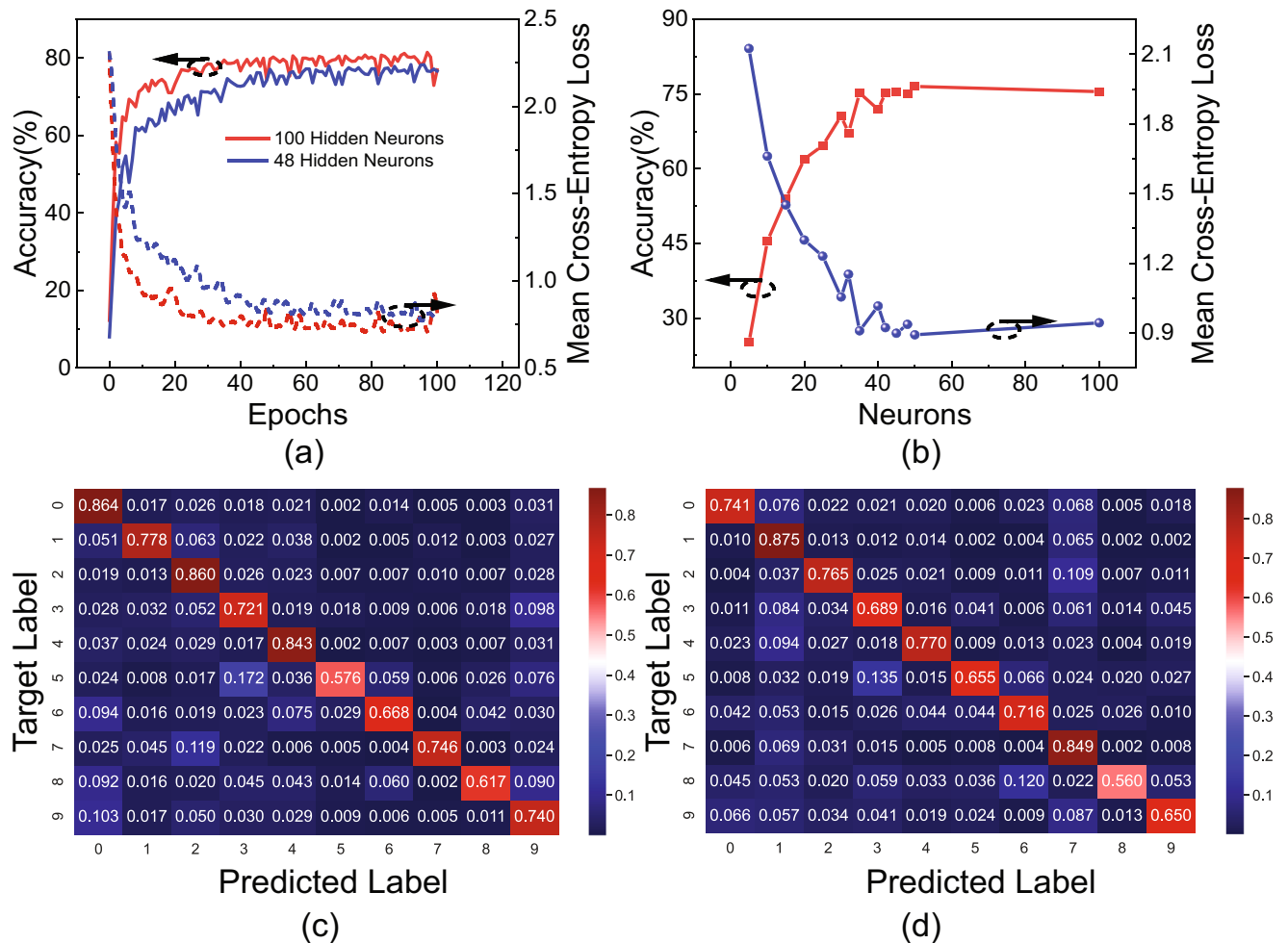


Fig. 7 Application of the proposed method to SVHN digit recognition. (a) Recognition accuracy (solid) and mean cross-entropy loss (MCEL) (dotted) as a function of number of training epochs with 100 and 48

neurons in the hidden layer. (b) Recognition accuracy (red) and MCEL (blue) as a function of neurons in the hidden layer. Confusion matrices for (c) 100 and (d) 48 neurons in the hidden layer

and fashion MNIST tests while the saturated MCEL is larger than them. This is largely due to the increased complexity in the SVHN dataset which considers the RGB compositions in an image. However, it is noticeable that the SVD method has not considerably affected the SLFN performance while reducing the number of neurons in the hidden layer. Test accuracy is depicted as a function of number of neurons in Fig. 7b, which validates that the system accuracy is maintained with the reduced number of neurons. However, there is a significant degradation in performance with less than 48 neurons. The confusion matrices with the original 100 neurons and those in a reduced number, 48, are depicted in Fig. 7c and d, respectively, in which a noticeable off-diagonal contribution is not observed. It is revealed that the optimally minimized SLFN architecture is capable of separating the output classes with confidence.

## 4.2 Performance comparison with previous studies

In order to make the distinction of the main purpose and performances of the methodology in this work, we have made comparisons between recent works and ours. Although several different algorithms have been reported in the literature related with optimization of neural network architectures, most of them have not focused on energy and area efficiency, but rather on expressive power with possibly complicated approximation function that can be obtained by the algorithm, toward a higher accuracy [50]. This is different from the main purpose of this work, the reduction of complexity in physical interconnection inside a hardware neural network architecture. Relatively similar topic was dealt in another literature [51], for a study on effects of neural network depth (number of hidden layers, not that of neurons in the single hidden layer as in this work) on the expressive power of the network. The focus is mainly made on the result of exponential increase in width, number of neurons in a hidden layer, in sacrifice of the reduction in depth, which does not provide any guideline or algorithmic approach for energy and area efficient hardware design of a neural network architecture. As compared with those reported without considerations in the hardware aspects, our methodology attempts to provide a practical algorithm for width reduction of an SLFN from a hardware application perspective. We have specifically chosen SLFN for our simulations due to the current research interest in these neural network architectures towards neuromorphic computing applications [52, 53]. In literature, the most widely used method for network optimization is the pruning technique in which the parameters of an existing network are systematically removed [54]. In general, the pruning technique starts with a pre-trained network, and then, removes the redundant parameters and repeats the training while maintaining a

**Table 1** Comparison between methods in the existing reports and in this work

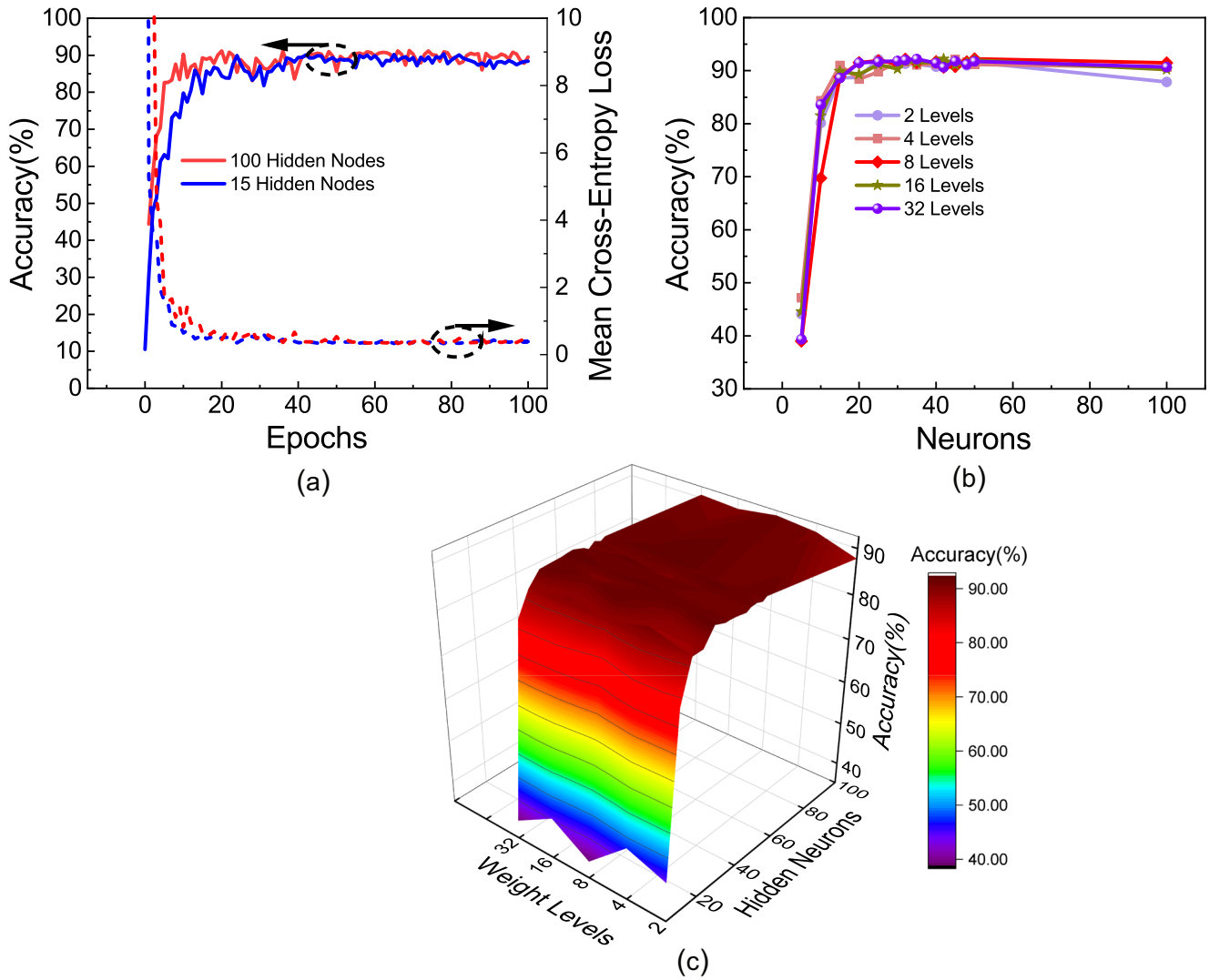
Reference	Network architecture	Methodology	Highlights	Comparison
Wen, et al.(2020) [34]	Recurrent neural network	Structured pruning method	20 times speedup on Penn TreeBank dataset	High complexity for hardware realization
Yeom, et al.(2021) [35]	Convolutional neural network	Layer-wise relevance propagation pruning method	Works on pre-trained models like VGG-16 and ResNet-18	Rather inefficient computation; speed is not considered
Floropoulos, et al.(2019) [36]	Fully connected, Convolutional neural networks	Complete vector quantization	15-20 times model compression for MNIST dataset	Computationally expensive implementation
Swaminathan, et al.(2020) [56]	Convolutional neural network	Sparse low rank method	Better compression as compared to truncated SVD models	Additional pruning and sparsification required
Guo, et al.(2019) [58]	Convolutional neural network	Low rank & sparse learning framework	Compresses VGGNet to 3.14% for CIFAR-10 dataset	High algorithm complexity; purely software; hardware realization not considered
This work	SLFN	SVD truncated with hardware considerations	Width reduction of 58% in MNIST dataset	Hardware realistic implementation with consideration of practical weight levels realized by synaptic devices; smaller array size and overhead circuitry for the integrated chip fabrication

similar model performance as the original unpruned model. A major drawback of some of the pruning techniques reported in literature is the nature of sparse neural network generated due to the removal of parameters. These pruned networks slow down the computation due to the uneven nature of memory access. The more structured pruning models also suffer from issues of pruning convergence, scheduling, and fine tuning. Recently, F. E. Fernandes Jr. et al., proposed the evolution strategy for the elimination of redundant filters from convolutional layers to reduce the computational complexity of the model [33]. Although the deep pruning with evolution strategy demonstrates plausible capabilities in terms of model compression and network performances, the algorithm introduces a major limitation as the method requires a population of individual CNN models of the same size to initiate the pruning process. Similarly, S. K. Yeom et al. proposed a layer-wise relevance propagation (LRP) method to compute the relevance scores for pruning the network. The authors identify a major drawback in their algorithm that the reduced network may not always be the smallest possible network and is strongly dependent on the chosen LRP variant [34]. H. Zhou et al. adopted a hierarchical pruning scheme using a self-organizing fuzzy neural network for nonlinear system modelling in industrial applications. Although the method is successful for non-linear system modelling, its applications is limited to fuzzy neural networks [55]. From a practical viewpoint, pruning adopted in these cases can effectively remove the neurons without considerable loss in accuracy in a given network fixed in the hardware sense but cannot contribute to the minimizing of network size from the initial design level for hardware realization. In comparison, a major achievement of our methodology is the reduction of the network model to the possible smallest representation for hardware realization. Another method for network optimization is the low-rank factorization technique [56]. Our approach is essentially based on this strategy but the optimization procedures are based on realistic hardware-oriented constraints in determining the reduced rank, unlike the existing one that is rather purely mathematical and can be suitable for software neural network, without deep considerations on synaptic device characteristics and number of neuron circuits. An existing technique is the network quantization approach where the number of bits needed to represent individual network parameters is reduced or quantized [36, 57]. Quantization is also applied to activations which saves additional memory allotments. A combination of low-rank compression and post-processing pruning strategy was recently reported [58]. However, the adapted algorithm is highly complex and optimised mainly for software-based implementation. In Table 1, network architecture, methodology, highlights, and comparisons are summarized in comparison between

the methodology in this work and the previously reported approaches for reducing the complexity of neural networks. Although the existing methods demonstrate plausible performances for network compression and high test accuracy, most of them are not free from the concerns about computational costs and additional overheads for hardware implementation: all these methodologies have been developed from software perspectives and have no realistic hardware-sense constraints. Hence, their hardware implementation would require additional neuron circuitry and very-large-scale crossbar array fabrication, both of which require time, cost, area, and energy in realization and operation. In comparison, in short, our proposed methodology has been developed specifically from a hardware point of view with requirements of minimal number of synaptic devices and neuron circuits. The reduced rank approximation is simple and effective from the neuromorphic hardware realization perspective.

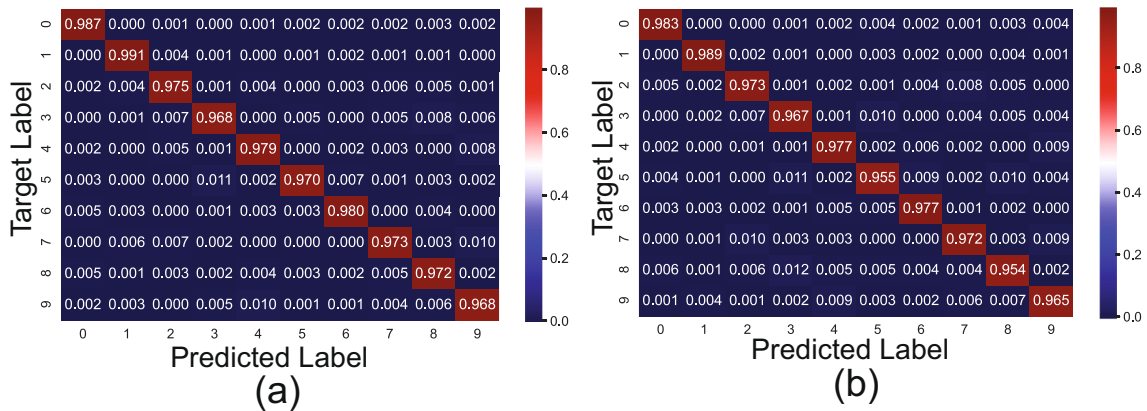
### 4.3 Synaptic transistor-based weight initialization

In order to evaluate the performance of our proposed method on hardware neuromorphic systems, we used a core-shell dual gate nanowire transistor as a synaptic device. Further details regarding the device and its synaptic capabilities has been summarised in our recent publication [28]. We have extracted the synaptic weights from the potentiation/depression characteristics of the device to initialise the neural network weights. The synaptic weights from the device have been chosen to form 2, 4, 8, 16, and 32 weight levels from the device. As discussed before in Section 3.1, these weight levels have been chosen based on practical hardware considerations. The neural network corresponding to each of these weight levels has been trained and optimised in a similar manner as explained in Algorithm 1. The initial network was trained using 100 neurons in the hidden layer whereas the optimised network ( $\gamma = 0.99$ ) after the application of the proposed method yielded 15 neurons in the hidden layer. Figure 8a shows the variation of classification accuracy and MCEL function as a function of the training epochs using 32-level device weights. It is evident from the figure that with the reduction in neurons there is no observable drop in the accuracy of the model. The reduction in number of neurons in the hidden layer is extremely significant from a hardware perspective since the CMOS implementation of neuronal circuits [3–5, 59, 60] require large capacitors for spike integration and pulse generation circuits which make these circuits bulky and consume more power. Due to the reduction in number of neurons in the hidden layer, the total number of synapses has also decreased from 79,400 to 11,910; which is highly advantageous for device fabrication. From a hardware perspective, such a reduction in the total number



**Fig. 8** Training with MNIST images. (a) Training accuracy (%) and mean cross-entropy loss as a function of the number of training epochs for 100 & 15 neurons in the hidden layer with 32 weight levels from a synaptic transistor device. (b) Test accuracy (%) variation with neurons

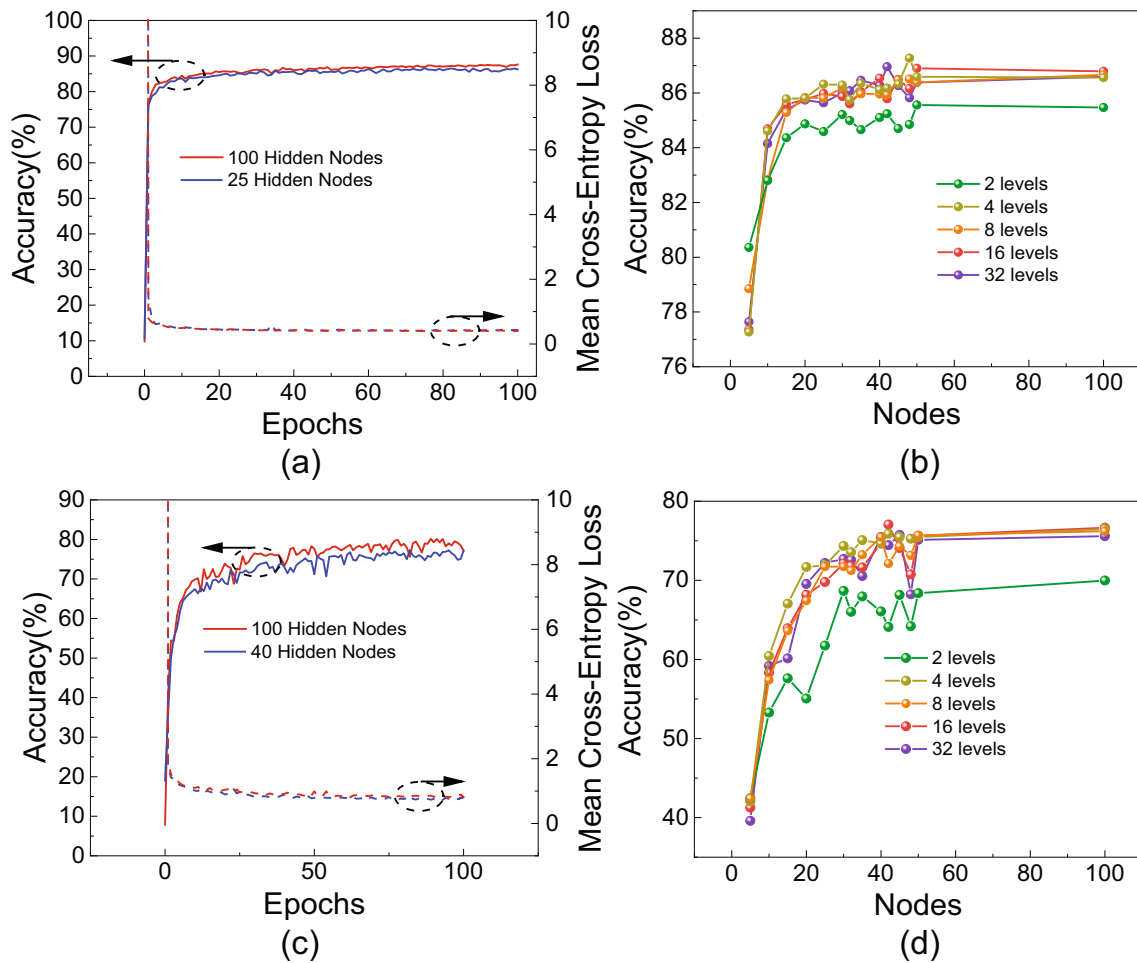
for various device weight levels. (c) 3D surface plot of the accuracy variation as a function of number of neurons in the hidden layer at different number of synaptic weight levels



**Fig. 9** Confusion matrix for the MNIST test data using 32 device weight levels for (a) 100 and (b) 15 neurons in the hidden layer

of synapses has a major advantage in terms of the synapse array space requirements and circuit integration density. The variation of test accuracy with the number of neurons in the hidden layer for various device weight levels is shown in Fig. 8b. We can observe that for all the weight levels, the test accuracy does not increase significantly beyond 15 neurons in the hidden layer. Additional neurons in the hidden layer are redundant in practice and do not contribute to the improvement of the model in terms of accuracy and generalization. The dependence of the accuracy on both the weight levels and neurons is shown in Fig. 8c, from which it is clear that the accuracy surface is a plane almost parallel to the space spanned by weight levels and neurons in the hidden layer beyond a threshold number of 15. The class separability of the reduced model based on 32 weight

levels has been compared with the initial model using the confusion matrix. The matrices for 100 and 15 neurons in the hidden layer are quite similar and there are no major off-diagonal elements which indicate the model capability to distinguish the output classes even with a reduced number of neurons in the hidden layer as demonstrated in Fig. 9a and b. We have also evaluated the performance of the SVD method on fashion MNIST and SVHN datasets similar to the evaluation done in Section 4.1. Figure 10a & b show the accuracies in fashion MNIST image recognition as a function of number of training epochs and that of neurons in the hidden layer. It is explicitly shown in Fig. 10a that the application of SVD has reduced the number of neurons in the hidden layer down to 25 from the original value of 100 without a significant accuracy drop for training with



**Fig. 10** Training with fashion MNIST and SVHN images. (a) Training accuracy and MCEL as a function of number of training epochs with 100 and 25 neurons in the hidden layer for fashion MNIST training. (b) Training accuracy as a function of number of neurons at different numbers of synaptic weights for fashion MNIST training. (c) Training

accuracy and MCEL as a function of number of training epochs with 100 and 40 neurons in the hidden layer for SVHN training. (d) Training accuracy as a function of number of neurons at different numbers of synaptic weights for SVHN training

**Table 2** Computational complexity comparison

	Implementation	Dataset	Number of neurons in the hidden layer	Number of synaptic devices
Conventional Model (uncompressed)	Fully software	MNIST	100	79,400
		fashion MNIST	100	79,400
		SVHN	100	308,200
Proposed model (SVD)	Hardware with synaptic devices	MNIST	100	79,400
		fashion MNIST	100	79,400
		MNIST 42	33,350	
	Fully software	fashion MNIST	30	23,820
		SVHN	15	11,910
		MNIST	15	11,910
Hardware with synaptic devices	fashion MNIST	25	19,850	
	SVHN	40	123,280	

fashion MNIST images. In Fig. 10b, the number of weights realized in a synaptic device is varied from binary to 32 levels and it is observed that it might be desirable to adopt multi-level synaptic device with no less than 4 levels for coping with SVD application. Figure 10c and d depict the accuracies in SVHN image recognition as a function of number of training epochs and that of neurons in the hidden layer. The application of SVD has an effect of reducing the number of neurons in the hidden layer down to 40 from the original value of 100 without experiencing a notable accuracy drop as shown in Fig. 10c. Also, it is revealed from Fig. 10d that multi-level synaptic device with 4 or more weights should be prepared for obtaining a high enough system accuracy. Reduction of the number of neurons is an advantage in terms of hardware implementation of a SLFN for saving area in realizing the neuromorphic chip. It is essential to figure out the optimally minimum number of neuron circuits maintaining the system accuracy, and thus, the proposed SVD method would be effective in reducing the model complexity and realizing area and power-efficient SLFN hardware. Table 2 summarises the reduction in computational parameters (number of synaptic weights) of the proposed model in comparison to the conventional (uncompressed) model. It is notable in Table 2 that both number of neurons in the hidden layer and number of synaptic devices are considerably reduced across different datasets for both software and hardware synaptic weight based implementations. Hence, the proposed SVD method is quite effective in reducing the model complexity and hence the resultant electrical circuitry for hardware realisation of the SLFN can be simplified considerably for practical neuromorphic chip realisation.

## 5 Conclusion

We have proposed a hardware realistic approach for determining the optimized number of neurons in the hidden layer in a SLFN. The methodology was applied to SLFN based on software (near-continuous) and hardware (discrete) synaptic weights. The application of the proposed method to MNIST dataset with hardware synaptic weights resulted in the reduction of neurons in the hidden layer to 15 from an initial value of 100. We have further verified the class separability of the reduced neuron models to be consistent with the initial model. The proposed method was further extended to fashion MNIST and SVHN datasets where the method was found effective in reducing the model complexity. The method also shines light on the mathematics of the hidden layer activation space. The simple and intuitive nature of the methodology makes it easily extendable for dealing with complicated networks with a large number of hidden layers and neurons.

**Algorithm 1** SVD.**Input:**

- $T$  train images  $\{X_1, X_2, \dots, X_T\}$
- Batch size=10, Learning rate=0.01
- Training epochs=100

**Training:**

- Initialise training with single hidden layer using large number (100) of neurons in the hidden layer

**while**  $epoch < 100$  **do****while**  $image$  in batch **do**

- Forward propagate through the network
- Compute errors using mean cross-entropy loss function
- Back-propagate using stochastic gradient descent algorithm

**end****end****Optimization:**

- Use trained model to recognise  $M$  test data images
- Compute the hidden layer activation matrix  $K \in \mathbb{R}^{M \times n}$
- Apply SVD on  $K$  to find diagonal matrix  $\beta \in \mathbb{R}^{M \times n}$
- From  $\beta$ , find  $\min(\sigma_g)$  such that equation (8) is satisfied

**if**  $\sigma_g$  satisfies equation (8) **then**

- Retrain model with  $g$  neurons in the hidden layer

**else**

- Check with  $g = g + 1$  for finding  $\sigma_g$

**Result:**

- Optimized number of hidden neurons given by  $g$

**Acknowledgements** This research was supported by National R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) (2021M3F3A2A01037927) and was also supported by the Gachon University Research Fund of 2019 (GCU-2019-0364).

**Declarations**

**Conflict of Interests** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

**References**

1. Ryu JH, Kim B, Hussain F, Ismail M, Mahata C, Oh T, Imran M, Min KK, Kim TH, Yang BD, Cho S, Park BG, Kim Y, Kim S (2020) Zinc tin oxide synaptic device for neuromorphic engineering. *IEEE Access* 8:130678–130686. <https://doi.org/10.1109/ACCESS.2020.3005303>
2. Cho S (2022) Volatile and nonvolatile memory devices for neuromorphic and processing-in-memory applications. *J Semicond Technol Sci* 22(1):30–46. <https://doi.org/10.5573/JSTS.2022.22.1.30>
3. Sourikopoulos I, Hedayat S, Loyez C, Danneville F, Hoel V, Mercier E, Cappy A (2017) A 4-fJ/Spike artificial neuron in 65 nm CMOS technology. *Front Neurosci* 11:123. <https://doi.org/10.3389/fnins.2017.00123>
4. Bayat FM, Prezioso M, Chakrabarti B, Nili H, Kataeva I, Strukov D (2018) Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nat Commun* 9(1):2331. <https://doi.org/10.1038/s41467-018-04482-4>
5. Lee JJ, Park J, Kwon MW, Hwang S, Kim H, Park BG (2018) Integrated neuron circuit for implementing neuromorphic system with synaptic device. *Solid State Electron* 140:34–40. <https://doi.org/10.1016/j.sse.2017.10.012>
6. Kim MH, Cho S, Park BG (2021) Nanoscale wedge resistive-switching synaptic device and experimental verification of vector-matrix multiplication for hardware neuromorphic application. *Jpn J Appl Phys* 60(5):050905. <https://doi.org/10.35848/1347-4065/abf4a0>
7. Goodfellow I, Bengio Y, Courville A (2016) Deep learning (MIT Press)
8. Merolla PA, Arthur JV, Alvarez-Icaza R, Cassidy AS, Sawada J, Akopyan F, Jackson BL, Imam N, Guo C, Nakamura Y, Brezzo B, Vo I, Esser SK, Appuswamy R, Taba B, Amir A, Flickner MD, Risk WP, Manohar R, Modha DS (2014) A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345(6197):668–673. <https://doi.org/10.1126/science.1254642>
9. Benjamin BV, Gao P, McQuinn E, Choudhary S, Chandrasekaran AR, Bussat JM, Alvarez-Icaza R, Arthur JV, Merolla PA, Boahen K (2014) Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc IEEE* 102(5):699–716. <https://doi.org/10.1109/JPROC.2014.2313565>
10. Necker A, Fok S, Benjamin BV, Stewart TC, Oza NN, Voelker AR, Eliasmith C, Manohar R, Boahen K (2019) Braindrop: a mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proc IEEE* 107(1):144–164. <https://doi.org/10.1109/JPROC.2018.2881432>
11. Schemmel J, Brüderle D, Grünbl A., Hock M, Meier K, Millner S (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: Proceedings of 2010 IEEE international symposium on circuits and systems, pp 1947–1950
12. Brown A, Furber S (2009) Biologically-inspired massively-parallel architectures - computing beyond a million processors. In: 2010 10th International conference on application of concurrency to system design ieeec computer society, Los Alamitos, CA, USA, pp 3–12. <https://doi.org/10.1109/ACSD.2009.17>
13. Qiao N, Mostafa H, Corradi F, Osswald M, Stefanini F, Sumislawska D, Indiveri G (2015) A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Front Neurosci* 9:141. 10.3389/fnins.2015.00141
14. Ma D, Shen J, Gu Z, Zhang M, Zhu X, Xu X, Xu Q, Shen Y, Pan G (2017) Darwin: a neuromorphic hardware co-processor based on spiking neural networks. *J Syst Archit* 77:43–51. <https://doi.org/10.1016/j.sysarc.2017.01.003>



15. Thakur CS, Molin JL, Cauwenberghs G, Indiveri G, Kumar K, Qiao N, Schemmel J, Wang R, Chicca E, Olson Hasler J, Seo JS, Yu S, Cao Y, Schaik AV, Etienne-Cummings R (2018) Large-scale neuromorphic spiking array processors: a quest to mimic the brain. *Front Neurosci* 12:891. <https://doi.org/10.3389/fnins.2018.00891>
16. Seo YT, Kwon D, Noh Y, Lee S, Park MK, Woo SY, Park BG, Lee JH (2021) 3-D And-type flash memory architecture with high- $\hat{\epsilon}$  gate dielectric for high-density synaptic devices. *IEEE Trans Electron Devices* 68(8):3801–3806. <https://doi.org/10.1109/TED.2021.3089450>
17. Bang S, Kim MH, Kim TH, Lee DK, Kim S, Cho S, Park BG (2018) Gradual switching and self-rectifying characteristics of Cu/ $\alpha$ -IGZO/p+-Si RRAM for synaptic device application. *Solid State Electron* 150:60–65. <https://doi.org/10.1016/j.sse.2018.10.03>
18. Ielmini D (2018) Brain-inspired computing with resistive switching memory (RRAM): Devices, synapses and neural networks. *Microelectron Eng* 190:44–53. <https://doi.org/10.1016/j.mee.2018.01.009>
19. Lee DK, Kim MH, Kim TH, Bang S, Choi YJ, Kim S, Cho S, Park BG (2019) Synaptic behaviors of HfO<sub>2</sub> ReRAM by pulse frequency modulation. *Solid State Electron* 154:31–35. <https://doi.org/10.1016/j.sse.2019.02.008>
20. Rasheed U, Ryu H, Mahata C, Khalil RMA, Imran M, Rana AM, Kousar F, Kim B, Kim Y, Cho S, Hussain F, Kim S (2021) Resistive switching characteristics and theoretical simulation of a Pt/a-Ta<sub>2</sub>O<sub>5</sub>/TiN synaptic device for neuromorphic applications. *J Alloys Compd* 877:160204. <https://doi.org/10.1016/j.jallcom.2021.160204>
21. Ambrogio S, Ciochini N, Laudato M, Milo V, Pirovano A, Fantini P, Ielmini D (2016) Unsupervised learning by spike timing dependent plasticity in phase change memory (PCM) synapses. *Front Neurosci*, vol 10, (56), <https://doi.org/10.3389/fnins.2016.00056>
22. Yan M, Zhu Q, Wang S, Ren Y, Feng G, Liu L, Peng H, He Y, Wang J, Zhou P, Meng X, Tang X, Chu J, Dkhil B, Tian B, Duan C (2021) Ferroelectric synaptic transistor network for associative memory. *Adv Electron Mater* 7(4):2001276. <https://doi.org/10.1002/aelm.202001276>
23. Shi Y, Oh S, Huang Z, Lu X, Kang SH, Kuzum D (2020) Performance prospects of deeply scaled spin-transfer torque magnetic random-access memory for in-memory computing. *IEEE Electron Device Lett* 41(7):1126–1129. <https://doi.org/10.1109/LED.2020.2995819>
24. Pal SK, Pramanik A, Maiti J, Mitra P (2021) Deep learning in multi-object detection and tracking: state of the art. *Appl Intell* 51(9):6400–6429. <https://doi.org/10.1007/s10489-021-02293-7>
25. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, pp 1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
26. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition
27. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition
28. Ansari MHR, Kannan UM, Cho S (2021) Core-shell dual-gate nanowire charge-trap memory for synaptic operations for neuromorphic applications. *Nanomaterials* 11(7):1773
29. Arai M (1993) Bounds on the number of hidden units in binary-valued three-layer neural networks. *Neural Netw* 6(6):855. [https://doi.org/10.1016/S0893-6080\(05\)80130-3](https://doi.org/10.1016/S0893-6080(05)80130-3)
30. Tamura S, Tateishi M (1997) Capabilities of a four-layered feedforward neural network: four layers versus three. *IEEE Trans Neural Netw* 8(2):251. <https://doi.org/10.1109/72.557662>
31. Fujita O (1998) Statistical estimation of the number of hidden units for feedforward neural networks. *Neural Netw* 11(5):851. [https://doi.org/10.1016/S0893-6080\(98\)00043-4](https://doi.org/10.1016/S0893-6080(98)00043-4)
32. Teoh EJ, Tan KC, Xiang C (2006) Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. *IEEE Trans Neural Netw* 17(6):1623–1629. <https://doi.org/10.1109/TNN.2006.880582>
33. Fernandes FEJ, Yen GG (2021) Pruning deep convolutional neural networks architectures with evolution strategy. *Inf Sci* 552:29–47. <https://doi.org/10.1016/j.ins.2020.11.009>
34. Wen L, Zhang X, Bai H, Xu Z (2020) Structured pruning of recurrent neural networks through neuron selection. *Neural Netw* 123:134–141
35. Yeom SK, Seegerer P, Lapuschkin S, Binder A, Wiedemann S, Müller K. R., Samek W (2021) Pruning by explaining: a novel criterion for deep neural network pruning. *Pattern Recogn* 115:107899. <https://doi.org/10.1016/j.patcog.2021.107899>
36. Floropoulos N, Tefas A (2019) Complete vector quantization of feedforward neural networks. *Neurocomputing* 367:55–63
37. Pal C, Pankaj S, Akram W, Acharyya A, Biswas D (2018) Modified Huffman based compression methodology for deep neural network implementation on resource constrained mobile platforms. *IEEE Int Symp Circuits Syst (ISCAS)*, pp 1–5, <https://doi.org/10.1109/ISCAS.2018.8351234>
38. Mellor J, Turner J, Storkey A, Crowley EJ (2021) Neural architecture search without training. In: Meila M, Zhang T (eds) *Proceedings of the 38th international conference on machine learning, proceedings of machine learning research*, vol 39, pp 7588–7598
39. Yu E, Cho S, Roy K, Park BG (2020) A quantum-well charge-trap synaptic transistor with highly linear weight tunability. *IEEE J Electron Devices Soc* 8:834–840. <https://doi.org/10.1109/JEDS.2020.3011409>
40. Kwon MW, Baek MH, Hwang S, Park K, Jang T, Kim T, Lee J, Cho S, Park BG (2018) Integrate-and-fire neuron circuit using positive feedback field effect transistor for low power operation. *J Appl Phys* 124(15):152107. <https://doi.org/10.1063/1.5031929>
41. Bartolozzi C, Benosman R, Boahen K, Cauwenberghs G, Delbrück T, Indiveri G, Liu SC, Furber S, Imam N, Linares-Barranco B, Serrano-Gotarredona T, Meier K, Posch C, Valle M (2016) *Neuromorphic systems*, Wiley encyclopedia of electrical and electronics engineering. Wiley, pp 1–22, <https://doi.org/10.1002/047134608X.W8328>
42. Blum A, Hopcroft J, Kannan R (2020) *Foundations of data science* (Cambridge University Press), <https://doi.org/10.1017/9781108755528>
43. Stewart GW (1973) Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Rev* 15(4):727
44. Teoh EJ, Tan KC, Xiang C (2006) Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. *IEEE Trans Neural Netw* 17(6):1623. <https://doi.org/10.1109/TNN.2006.880582>
45. Marti D, Rigotti M, Seok M, Fusi S (2016) Energy-efficient neuromorphic classifiers. *Neural Comput* 28(10):2011–2044
46. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324. <https://doi.org/10.1109/5.726791>
47. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, D'Alché-Buc F, Fox E, Garnett R (eds) *Advances in neural information processing systems* 32, pp 8024–8035.(Curran Associates, Inc.)

48. Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning
49. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning 2011
50. Cohen N, Sharir O, Shashua A (2016) On the Expressive Power of Deep Learning: A Tensor Analysis. In: Feldman V, Rakhlin A, Shamir O (eds) 29th Annual conference on learning theory, proceedings of machine learning research, vol 49, pp 698–728, PMLR, Columbia University, New York
51. Yarotsky D (2017) Error bounds for approximations with deep ReLU networks. *Neural Netw* 94:103. <https://doi.org/10.1016/j.neunet.2017.07.002>
52. Subbulakshmi Radhakrishnan S, Sebastian A, Oberoi A, Das S, Das S (2021) A biomimetic neural encoder for spiking neural network. *Nat Commun* 12(1):2143. <https://doi.org/10.1038/s41467-021-22332-8>
53. Yan Z, Chen J, Hu R, Huang T, Chen Y, Wen S (2020) Training memristor-based multilayer neuromorphic networks with SGD, momentum and adaptive learning rates. *Neural Netw* 128:142–149
54. Mohammed MF, Lim CP (2017) A new hyperbox selection rule and a pruning strategy for the enhanced fuzzy minâmax neural network. *Neural Netw* 86:69–79
55. Zhou H, Zhang Y, Duan W, Zhao H (2020) Nonlinear systems modelling based on self-organizing fuzzy neural network with hierarchical pruning scheme. *Appl Soft Comput* 106516:95. <https://doi.org/10.1016/j.asoc.2020.106516>
56. Swaminathan S, Garg D, Kannan R, Andres F (2020) Sparse low rank factorization for deep neural network compression. *Neurocomputing* 398:185–196
57. Huang C, Liu P, Fang L (2021) MXQN: mixed quantization for reducing bit-width of weights and activations in deep convolutional neural networks. *Appl Intell* 51(7):4561. <https://doi.org/10.1007/s10489-020-02109-0>
58. Guo K, Xie X, Xu X, Xing X (2019) Compressing by learning in a low-rank and sparse decomposition form. *IEEE Access* 7:150823. <https://doi.org/10.1109/ACCESS.2019.2947846>
59. Indiveri G, Chicca E, Douglas R (2006) A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans Neural Netw* 17(1):211–221. <https://doi.org/10.1109/TNN.2005.860850>
60. Wu X, Saxena V, Zhu K, Balagopal S (2015) A CMOS spiking neuron for brain-inspired neural networks with resistive synapses and in situ learning. *IEEE Trans Circuits Syst II: Express Br* 62(11):1088–1092. <https://doi.org/10.1109/TCSII.2015.2456372>
61. Yu E, Cho S, Roy K, Park BG (2020) A quantum-well charge-trap synaptic transistor with highly linear weight tunability. *IEEE J Electron Devices Soc* 8:834–840. <https://doi.org/10.1109/JEDS.2020.3011409>
62. Cho Y, Lee JY, Yu E, Han JH, Baek MH, Cho S, Park BG (2019) Design and characterization of semi-floating-gate synaptic transistor. *Micromachines* 10(1). <https://doi.org/10.3390/mi10010032>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Kannan Udaya Mohanan received his Ph.D. degree in applied physics from Indian Institute of Technology (IIT) Hyderabad, India, in 2019. He worked as a Senior Project Scientist at IIT Delhi, India, during 2019–2021, where he mainly worked on the development of resistive-switching devices with low-power operation. He is currently working as an Assistant Professor at the Department of Electronic Engineering, Gachon University, Seongnam, Republic of Korea. His current research focuses on novel electron devices and efficient algorithms for hardware realization of neuromorphic computing architecture.



Seongjae Cho received the B.S. and the Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Republic of Korea, in 2004 and 2010, respectively. He worked as an Exchange Researcher at the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan, in 2009. Also, he worked as a Postdoctoral Researcher at Seoul National University in 2010 and at Stanford University, CA, USA, from 2010 to 2013. He joined the Department of Electronic Engineering, Gachon University, Seongnam, Republic of Korea, in 2013, where he is currently working as an Associate Professor. His current research interests include emerging memory technologies, advanced nanoscale CMOS devices, group-IV photonic devices, memory cells for neuromorphic and memory-centric processor technologies. He received the Haedong Young Engineer Award from the Institute of Electronics and Information Engineers (IEIE) of Korea in 2011 and the Minister Award for Semiconductor Fundamentals Innovation from the Korean Ministry of Science and ICT in 2021.

**Kannan Udaya Mohanan** received his Ph.D. degree in applied physics from Indian Institute of Technology (IIT) Hyderabad, India, in 2019. He worked as a Senior Project Scientist at IIT Delhi, India, during 2019–2021, where he mainly worked on the development of resistive-switching devices with low-power operation. He is currently working as an Assistant Professor at the Department of Electronic Engineering, Gachon University, Seongnam, Republic of

**Seongjae Cho** received the B.S. and the Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Republic of Korea, in 2004 and 2010, respectively. He worked as an Exchange Researcher at the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan, in 2009. Also, he worked as a Postdoctoral Researcher at Seoul National University in 2010 and at Stanford University, CA, USA, from 2010 to



**Byung-Gook Park** received the B.S. and the M.S. degrees in electronic engineering from Seoul National University in 1982 and 1984, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1990. From 1990 to 1993, he was with AT&T Bell Laboratories, where he contributed to the development of 0.1- $\mu$ m CMOS and its characterization. From 1993 to 1994, he was with Texas Instruments, developing 0.25- $\mu$ m CMOS. In

1994, he joined Seoul National University. He led the Inter-university Semiconductor Research Center (ISRC), Seoul National University, as the Director from 2008 to 2010. He received Best Teacher Award from Seoul National University in 1997, Education Award from College of Engineering, Seoul National University in 2006, Haedong Academic Research Award from IEIE in 2008, the Minister Award for Nano Research Innovation from the Korean Ministry of Science, ICT and Future Planning in 2013. He is an IEIE Life Member, Member of the National Academy of Engineering of Korea, and IEEE Fellow. He received the Greatest National Award from the Ministry of Science and ICT in 2022 for his incessant contributions in semiconductor science and technology.